Zusammenfassung DASB

Data Science Basics

Maurin D. Thalmann

24. Januar 2020

Inhaltsverzeichnis

1 Learning Objectives

Introduction to Data Science

- Define the term „Data Science“ and differentiate it from similar concepts such as big data, data mining, OLAP, predictive modeling, data analysis
- Describe why and how value is created from data, especially in form of processes
- Describe the skills and activities of a data science professional
- Describe possible applications of data science

Intro to R and Exploratory Data Analysis (EDA)

- Define the term „Exploratory Data Analysis“ and explain why it is a fundamental part of making data science
- Give a short introduction to R and present its main features
- Indicate the key actors and actions that compose EDA and how they are related to the data sources considered
- Indicate different ways of importing data into R for analysis and how to check for their quality
- Describe the main operations available in R (through libraries) to manipulate data and prepare it for the analysis

ggplot2 and its usage in EDA

- Define the role of graphical representations in the „Exploratory Data Analysis“ and explain why it is a fundamental tool to make it effective
- Give a short introduction to ggplot2 and its features
- Ddescribe the main way of working of ggplot2 and how to compose complex graphs (e.g.: representing more than 3 variables in a single graph, for multidimensional data)
- Explain the role of the facets and how they can support the representation of „partial“ relationships
- Indicate the role of the coordinates in the graph usage for EDA

Intro to Forecasting

- Understand and explain the concept of forecasting
- Present the basic approach, either using the theory or by example
- Present a linear model and describe its basic characteristics
- Indicate how EDA can support and scaffold the model identification
- Apply basic forecasting in R, given the datasource

2 Introduction to Data Science

2.1 Introductory Stizzle

- More is different - with different quantity comes different quality, although „more“ is always dependent on the point of view.
- Data volume nowadays is exploding (276.12 billion GB of digital data)
- Data Science tries to close the gap between „Data *available* to an organization“ and „Percent of data an organization can *process*“.

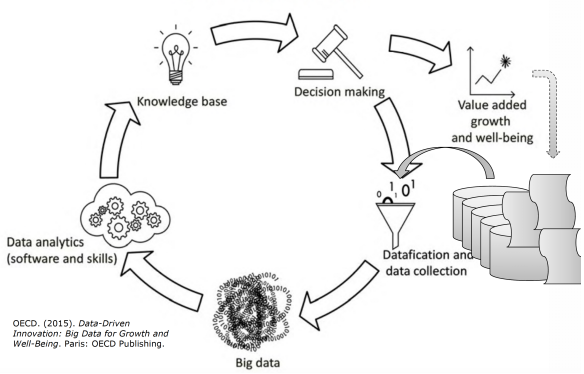


Abbildung 1: Beautiful illustration of the classic data value cycle

„**Data Science** is the extraction of actionable knowledge directly from data through a discovery, or hypothesis formulation and hypothesis testing“. Data Scientists generate knowledge from big data.

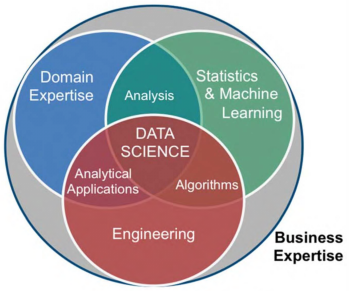


Abbildung 2: Skills needed in Data Science

2.2 A Global View on Data Science

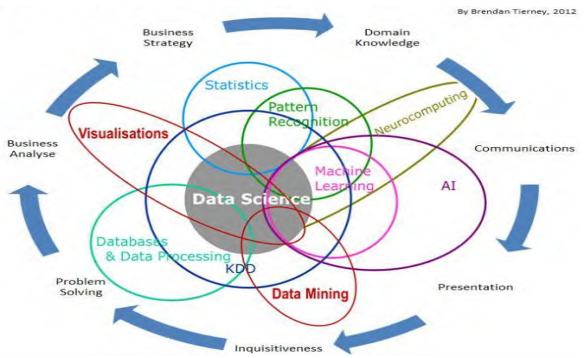


Abbildung 3: Data Science is multidisciplinary

- **AI**: programs that perform tasks resembling humans (learn and reason)
- **ML**: algorithms to learn from that data without explicit programming
- **DL**: subset of ML using artificial neural networks for treating vast amount of data (big data)
- **Data Science**: spans the collection, management, analysis and interpretation of large amounts of data with a wide range of applications → make informed decisions based on what was learned
- **EDA** (exploratory data analysis): extract insight from data (outside AI, human based)

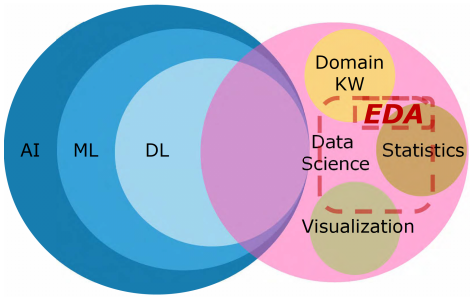
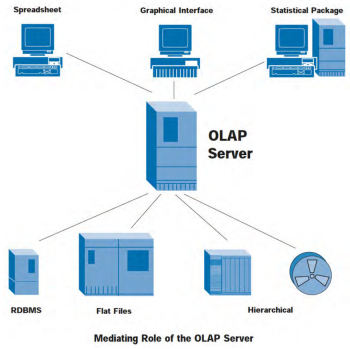


Abbildung 4: AI / ML / DL and Data Science

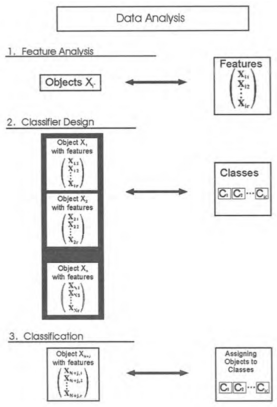
2.3 Different Concepts in Data Analytics

2.3.1 1993 - Online Analytical Processing (OLAP)



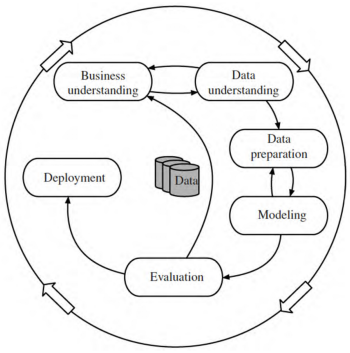
- It's online, not batch (interactive, not programmed in COBOL)
- The OLAP system should access the data required to perform the indicated analysis
- OLAP tools empower useranalysts to earily perform multi-dimensional analysis, which previously have been avoided because of their perceived complexity.

2.3.2 1998 - Fuzzy Data Analysis



- Data analysis can be defined as **search for structure** in data
- In data analysis, objects are considered which are described by some attributes
- Most of the traditional methods for data analysis assume that patterns to be detected are two-valued
- Whenever this is not the case, the relationship between data and classes becomes gradual
  → Fuzzy Classification

2.3.3 2005 - Data Mining



- Very similar concept to data science
- Machine Learning (modeling) is the technical core of practical data mining applications
- Data *Mining* is a business process related to *value* (finding the metaphorical gold nugget)
- The lifecycle of a data mining project is defined by the CRISP-DM reference model

2.3.4 2013 - Predictive Modeling

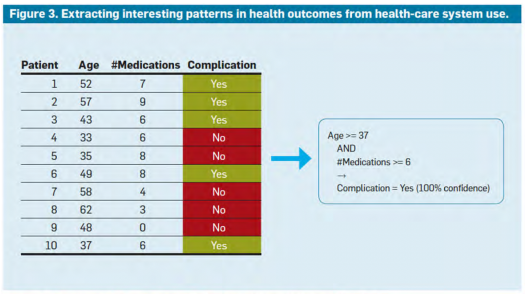


Figure 3. Extracting interesting patterns in health outcomes from health-care system use.

Patient	Age	#Medications	Complication
1	52	7	Yes
2	57	9	Yes
3	43	6	Yes
4	33	6	No
5	35	8	No
6	49	8	Yes
7	58	4	No
8	62	3	No
9	48	0	No
10	37	6	Yes

Age >= 37
AND
#Medications >= 6
→
Complication = Yes (100% confidence)

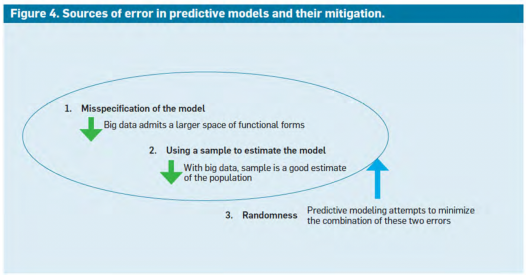
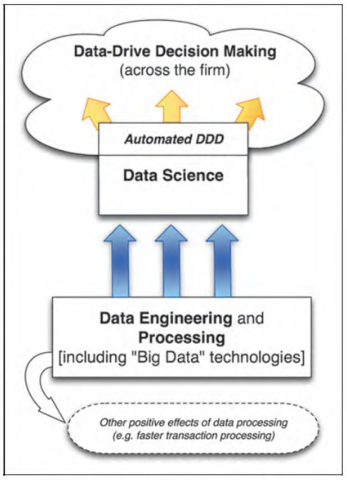


Figure 4. Sources of error in predictive models and their mitigation.

1. Misspecification of the model
Big data admits a larger space of functional forms
2. Using a sample to estimate the model
With big data, sample is a good estimate of the population
3. Randomness
Predictive modeling attempts to minimize the combination of these two errors

- A common epistemic requirement in assessing whether new knowledge is actionable for decision making in its *predictive power*, not just its ability to explain the past.
- The requirement on predictive accuracy on observations that will occur in the future is a key consideration in data science.

2.3.5 2013 - Data Driven Decision Making



- Data science involves much more than just data-mining algorithms.
- Successful data scientists must be able to view business problems form a *data perspective*.

3 Intro to R and Exploratory Data Analysis (EDA)

3.1 Intro to R

- **R** ist a software and programming language for statistical applications and graphics
- It's an implementation of the programming language S, with some extensions
- It provides functions for linear/non-linear modelling, classical statistical tests, time-series analysis, classification, clustering etc.
- **RStudio** is an integrated development environment (IDE) for R

3.1.1 Basic Operation in R

- Assignment:
  `my_var <- 46, nr <- 38`
  (You can also use `copy my_var = 46`, but it is not suggested)
- Printing (explicit and implicit): every line with only a name of an object
  Implicit: `my_var`
  Explicit using functions like print: `print(my_var)`
- Arithmetic operators:
  `4 + 5`
  `2 ^ 3 - 7 %% 3 - 2 ^ (3 - 7 %% 3)` etc. etc. etc.
- Control flow, conditionals, loops can be used as in any other language
  `for (i in 1:10) print(i)`
  `while (nr < 42) print(,,We have ‘‘+str(nr)+,, : still space...‘‘); nr=nr+1`

3.2 What is EDA

- Exploratory Data Analysis is the
  - critical process of performing initial investigations on data so as to
    - discover patterns
    - spot anomalies
    - test hypothesis
    - check assumptions
  - with the help of **summary statistics** and **graphical representations**

The use of a statistical model is not mandatory, in this step, even if it is normally adopted to test hypotheses and check assumptions. For anomalies and patterns discovery, correlations (represented into the correlation matrix) and distribution (represented with box and whisker diagrams) are the usual approaches.

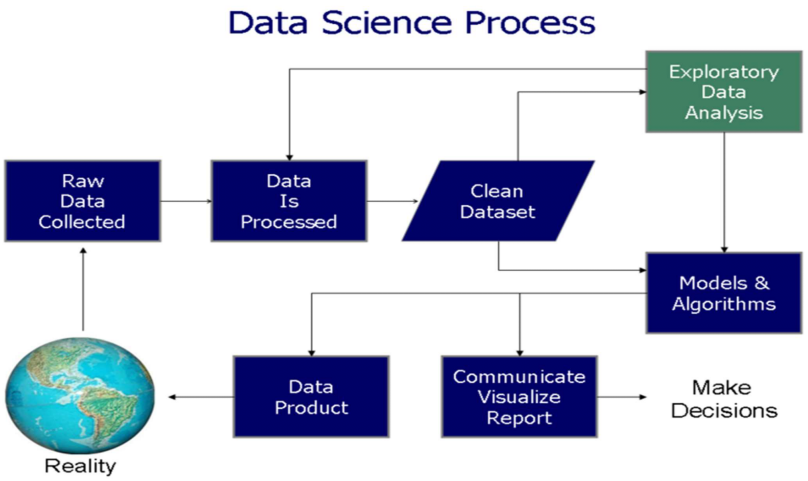


Abbildung 5: EDA in the context of Data Science

3.2.1 EDA process „in practice“

- EDA is an iterative cycle
- Process:
  - Generate questions about the data
  - Search for answers by visualizing, transforming and modelling the data
  - Use the insights to refine the questions and/or generate new questions
- Start with many ideas...
- some will be evidently useless and other will end up in a dead end
- a few of them will help to have a better understanding of the data/process
- repeat with generating other ideas until the insight is enough
- then you can apply the model to the data for forecasting and process explanation
- EDA also allows you to verify/test the quality of your data

3.3 Data Import / Transform / Link

3.3.1 Data Import

- R provides importing functions, depending on the format of the data source:
  - CSV / TSV / delimited texts: `read_csv()` & `read_csv2()` (semicolon), `read_tsv()`, `read_delim()`
  - Fixed width columns: `read_fwf()` & `read_table()`
  - Logs (e.g. webserver logs): `read_log()`

These are all similar in terms of parameters, examples:

```
read_delim(file, delim, quote = "\"", escape_backslash = FALSE, escape_double = TRUE,
col_names = TRUE, col_types = NULL, locale = default_locale(), na = c("", "NA"),
quoted_na = TRUE, comment = "", trim_ws = FALSE, skip = 0, n_max = Inf, guess_max =
```

3.3.2 Data Transform (example dataset)

Example dataset loaded from: *Auto MPG Data Set*

```
> library(tidyverse)
> DS <- read.table("C:/Users/JumpStart/Downloads/auto-mpg.data",
quote="\"", comment.char="")
> names(DS) <-
c("mpg","cylinders","displacement","horsepower","weight","acceleration","
model year","origin","car name")
> DS <- as_tibble(DS); attach(DS); View(DS)
```

3.3.3 Data Transform

- Filter rows with `filter()`
  - subset observations based on their values
    `filter(DS, weight > 3500)` or `filter(DS, DS$weight > 3500)`
- Arrange rows with `arrange()`
  - changes the order of rows
    `arrange(DS, desc(DS$acceleration))`
- Select columns with `select()`
  - Subset columns with position and/or negative condition (by name or index)
    `select(DS, -1)`
    `select(DS, 2, 3, 4)`
    `select(DS, c(mpg, 5))`
- Add new variables with `mutate()`
  - Allow to create new dimension(s)
    `mutate(DS, a4w = acceleration/weight, pre75 = as.numeric(DS$'model year' < 75))`
- Grouped summaries with `summarise()`
  - Allows to compute aggregate metrics
    `summarise(group_by(DS, 'model year'), mean_display = mean(displacement), sd_displ = sd(displacement), n=n()`

- All verbs work similarly:
  - The first argument is a dataframe
  - The subsequent arguments describe what to do with the data frame, using the variable names (without quotes)
  - The result is a new data frame

3.3.4 Data „Linkage“

Some prices of cars of the 70's: *70's Cars*.
This data is imported into a new tibble and we will try to use it for creating another View for exploring the relationship between cost, year, power and mpg.

```
> car_cost <-read_delim("C:/Users/JumpStart/switchdrive/__/DASB/SW02/car_197x_costs_2.csv",
    delim = "-",
    escape_double = FALSE,
    na = "NA",
    trim_ws = TRUE,
    col_types = cols(`Car Name` = col_factor(),
    Price = col_character(),
    Location = col_character(),
    `Matriculation Year` = col_integer()))
> car_cost <- as_tibble(car_cost)
> car_cost <- mutate(car_cost, Location = sapply(Location,as.factor))
> car_cost <- mutate(car_cost, Price = as.integer(str_replace(str_sub(Price, 2, -1),",",""")))
> View(car_cost)
```

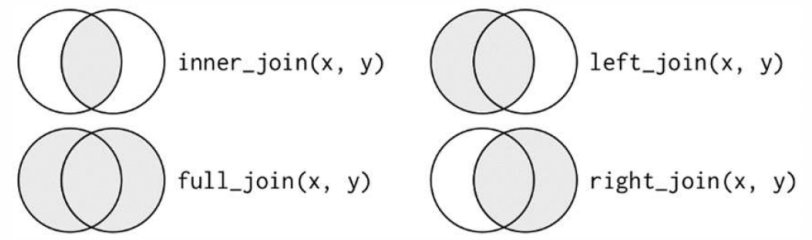


Abbildung 6: R functions to link data frames with each other

There are different types of R functions for linking tibbles/dataframes (~datasets)

- **Mutating** joins allows to combine variable from two tables
  - `inner_join(x, y, by = keys)`
    matches pairs of observations whenever their keys are equal
  - Outer join keeps observations that appear in at least one of the tables. There are three types of outer joins:
    - `left_join(x, y, by = keys)`
      keeps all observations in x
    - `right_join(x, y, by = keys)`
      keeps all observations in y
    - `full_join(x, y, by = keys)`
      keeps all observations in x and y
- **Filtering** joins allow to restrict the set of row from the 1^{st} table
  - `semi_join(x, y, by = keys)`
    keeps all observations in x that have a match in y
  - `anti_join(x, y, by = keys)`
    drops all observations in x that have a match in y
- **Set Operations** (work on complete rows, comparing each variable values. Inputs needs the same variables, for treating observations like sets)
  - `intersect(x, y)`
    return only observations in both x and y
  - `union(x, y)`
    return unique observations in x and y
  - `setdiff(x, y)`
    return observations in x, but not in y

3.3.5 Applying data loading/transforming/linking

- Find one or more data sources you are interested into (UCI ML Repository / Kaggle)
- Start the EDA „mindset“:
  1. What I believe about the data / what I would like to discover
     $\rightarrow$ Make hypotheses and formulate assumptions
  2. Think how this can be supported by data $\rightarrow$ experiment
  3. Run the experiment to verify
  4. Acquire new knowledge that can help you improve step 1
- Repeat the process at least 3 times
  (Optional: short report of the process you followed, document it in a short presentation)

4 ggplot2 and its usage in EDA

4.1 ggplot2 - Intro

- Used with the library „ggplot2“
  `library(ggplot2)`
- Simple usage example:
  `plot <- ggplot(data = mpg, mapping = aes(x = mpg$hwy))`
  plot + geom_histogram()
- ggplot 2 package:
  - produces layered statistical graphics.
  - uses an underlying „grammar“ to build graphs layer-by-layer rather than providing premade graphs
  - is easy enough to use without any exposure to the underlying grammar, but is even easier to use once you know the grammar
  - allows the user to build a graph from concepts rather than recall of commands and options

4.2 ggplot2 - layer after layer

- The initial call to ggplot does create the graph, but no content:
```
ggplot(data = mpg)
ggplot(data = mpg, mapping = aes(x = mpg$hwy))
ggplot(data = mpg, mapping = aes(x = mpg$hwy, y = mpg$dipl))
```
- We need to add a layer by using + (or collect the partial graph in an object)
```
p <- ggplot(data = mpg)
p + geom_histogram(mapping = aes(x = mpg$hwy))
```
- Parameters set into the ggplot() call are inherited by every other layer, but can be overridden, if needed
```
p <- ggplot(data = mpg)
p <- p + geom_histogram(mapping = aees(x = mpg$hwy))
p
p + geom_violin(mapping = aes(y = mpg$displ))
```

4.3 Elements of grammar of graphics

- **Data**: variables mapped to aesthetic features of the graph.
- **Geoms**: objects / shapes on the graph.
- **Stats**: statistical transformations that summarize data (e.g. mean, confidence intervals).
- **Scales**: mappings of aesthetic values to data values. Legends and axes visualize scales.
- **Coordinate systems**: the plane on which data is mapped on the graphic.
- **Faceting**: splitting the data into subsets to create multiple variations of the same graph (paneling).

4.4 Aesthetics

- Visual properties of objects on the graph (depends on the geom*)
- Commonly used aesthetics:
  x positioning along x-axis
  y positioning along y-axis
  color color of objects; for 2D-objects: the color of the object's outline (compare to fill below)
  fill fill color of objects
  linetype how lines should be drawn (solid, dashed, dotted etc.)
  shape shape of markers in scatter plots
  size how large objects appear
  alpha transparency of objects (value from 0 [transparent] to 1 [opaque])

4.5 Geoms

- Geometric shapes produced for the plot.
  - `geom_bar()`: bars with bases on the x-axis
  - `geom_boxplot()`: boxes and whiskers
  - `geom_density()`: density plots
  - `geom_histogram()`: histogram
  - `geom_line()`: lines
  - `geom_point()`: points (scatterplot)
  - `geom_rug()`: rug plot (2D display with the two 1D marginal distributions)
  - `geom_smooth()`: smoothed conditional means
  - `geom_text()`: text

4.6 Stats & Scales

- **Stats** → statistically transform data, usually as some form of summary (e.g.: mean, standard deviation, confidence interval etc.). Each stat function is associated with a default geom, so no geom is required for shapes to be rendered.
- **Scales** → control how a plot maps data values to the visual values of an aesthetic
  - `scale_color_manual()`:
    define an arbitrary color scale, by scecifying each color manually
  - `scale_color_hue()`:
    define an evenly-spaced color scale, by specifying a range of hues and the number of colors on a scale
  - `scale_shape_manual()`
    define an arbitrary shape scale, by specifying each shape manually
- Scales can also be applied to axes:
  - `scale_x_continuous()`
    map continuous to visual values - `scale_x_discrete()`
  - `scale_x_date(labels = date_format(„%m/%d“), breaks = date_breaks(„2 weeks“))`:
    treat x values as dates - scale_x_datetime()
  - `scale_x_log10()`:
    plot x on log10 scale
  - `scale_x_reverse()`:
    reverse direction of x-axis
  - `scale_x_sqrt()`:
    plot x on square root scale

4.7 ggplot2 - Initial Example

```
ggplot(diamonds, aes(x=carat, y=price, color=cut)) +
  geom_point() +
  scale_color_manual(
  values=c("red", "yellow", "green", "blue", "violet")
) +
scale_y_continuous(
  breaks=c(0,2500,5000,7500,10000,12500,15000,17500),
  labels=c(0,2.5,5,7.5,10,12.5,15,17.5),
  name="price(thousands of dollars)"
)
```

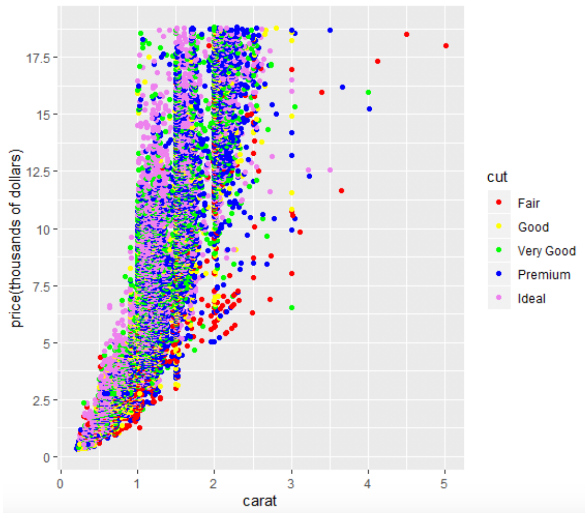


Abbildung 7: Rendered graph for Initial Example

4.8 Coordinate System & Faceting

- **Coordinate system**: set the planes for objects positioning on the plot
- Cartesian is the most diffused and used by default. Other coordinate systems would be:
  - polar
  - flipped Cartesian
  - fixed-ratio Cartesian
  - Map projections (mercatore, lagrange)
- **Faceting**: divide the graph into subgraphs (panels), using variable values
  - `facet_wrap(~{vars})`:
    create a multirow panel of plots, `{vars}` represents a list of splitting variables, separated by +
  - `facet_grid({vars}~{vars})`:
    row-splitting variable before, the column-splitting variable after
- Other aspects include the teming, position adjustment, labels and legends, zooming and the graph savings (ggsave)

4.9 ggplot2 - Second Example

```
ggplot(diamonds, aes(x=carat, y=price, color=cut)) +
  geom_point() +
  facet_grid(clarity~color) +
  scale_color_manual(values=c("red", "yellow", "green", "blue", "violet")) +
  scale_y_continuous(breaks=c(0,2500,5000,7500,10000,12500,15000,17500),
  labels=c(0,2.5,5,7.5,10,12.5,15,17.5),name="price(thousands of dollars)")
```

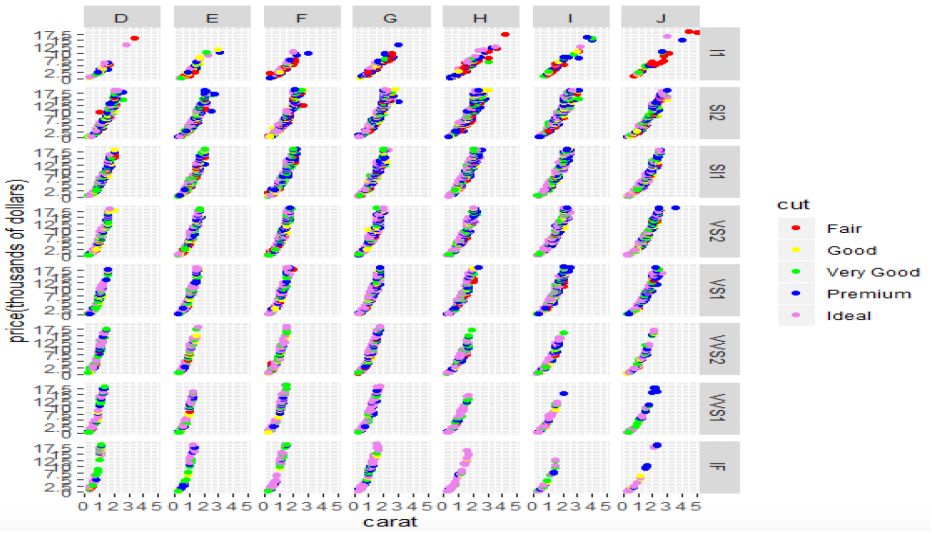


Abbildung 8: Rendered graph for Second Example

5 Shiny

Machi de wenni no Loscht han. (Wenn de Text am Schloss no do esch, hani kei Loscht meh gha)

6 Intro to Forecasting

6.1 Indication

- Forecast: the ability to use a set of variables for predicting another one
- Predictors $\rightarrow F(x) \rightarrow$ Output
- Can assume two forms:
  - Continuous Output $\rightarrow$ Regression
  - Discrete (Categorical) Output $\rightarrow$ Classification
- It uses similar approaches, but with a different result
- The function $F(x)$ is called the model. Can be used for two goals:
  - Pure prediction: care only for the forecast
  - Interpretation: care of the existing relationship, to better understand the phenomena

6.2 Linear Modeling

6.2.1 Modeling

- Simplest form of a model is the Linear Model
- With one predictor (bidimensional space) represent the equation of a line:
  $y = F(x) = \beta_0 + \beta_1 \cdot x$

$$Y \approx \beta_0 + \beta_1 \cdot X$$

$$e_i = y_i - \hat{y}_i$$

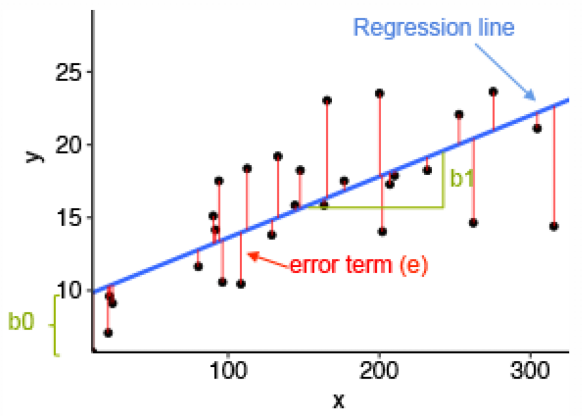
$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

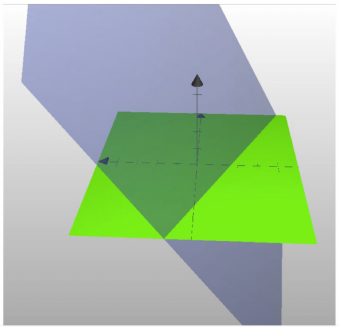
$$\bar{y} \equiv \frac{1}{n}\sum_{i=1}^{n} y_i$$

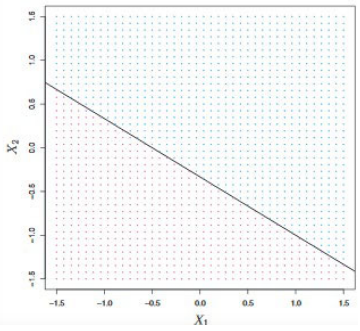
$$\bar{x} \equiv \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\text{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2$$



- With more predictors (multidimensional space or Hyperspace),
  the model can be seen as a hyperplane:
  $y = F(x) = \beta_0 + \beta_1 \cdot + \beta_2 \cdot x_2 + \cdots + \beta_n \cdot x_n$





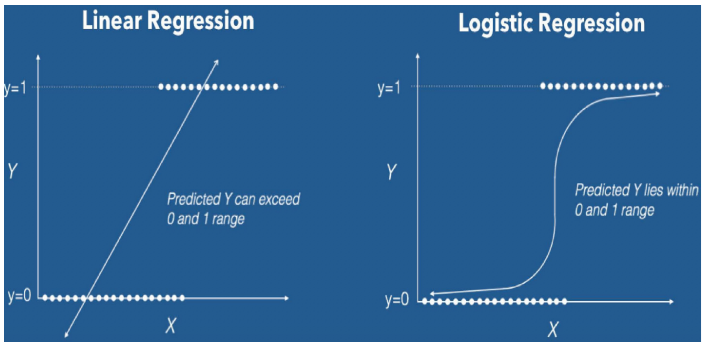
6.2.2 Model Identification

Following steps need to be done

- Define the predictors set and the output variables (also using correlation)
  - $x = \ldots, y = \ldots$
- Create the model
  - `model <- lm(y ~ x, data = d)`
- Inspect the model
  - Parameters $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$
- Verify that the model is „trustable“
  - P-values of each β_a, everyone should be under 0.05 (5%)

6.2.3 Prediction

- It is natural to interpret the model for regression:
  - Given a single set of predictor values X (point) $\rightarrow$ compute the value of the output (y)
- but what about for classification?
  - We need to define a way to transform the output of the linear model to our output categories
  - This is done by the logistic function
- How to do predictions:
  - `Pred <- predict(model, data = d2)`



6.2.4 Model Evaluation

- We need to evaluate models based on performances
  - Regression $\rightarrow$ the sum of squared errors (RSS $\rightarrow$ residual sum of squares)
  - Classification $\rightarrow$ the misclassification error rate
- Every model with more predictors has a higher predictive power...
  ...but, we don't want to learn the noise on the data!
- Create an independent test set
  - Use most part (e.g. 70%) of the data for training purposes
  - Use the rest for the model performance estimation $\rightarrow$ Test Error Rate